Microsoft

April 7-8, 2025
sambaXP Conference

Microsoft Interoperability Track

Microsoft

SMB in Windows Server 2025 and Beyond

sambaXP 2025, April 7-8

Raymond Wang, Genghis Karimov
& Dan Cuomo

SMB Product Group, Microsoft

Microsoft

SMB in Windows Server

2025 and vNext

Microsoft

SMB in Windows Server 2025 and vNext

Raymond Wang

Principal Software Engineering Lead

Genghis Karimov

Principal Software Engineer

Dan Cuomo

Principal Program Manager

Agenda

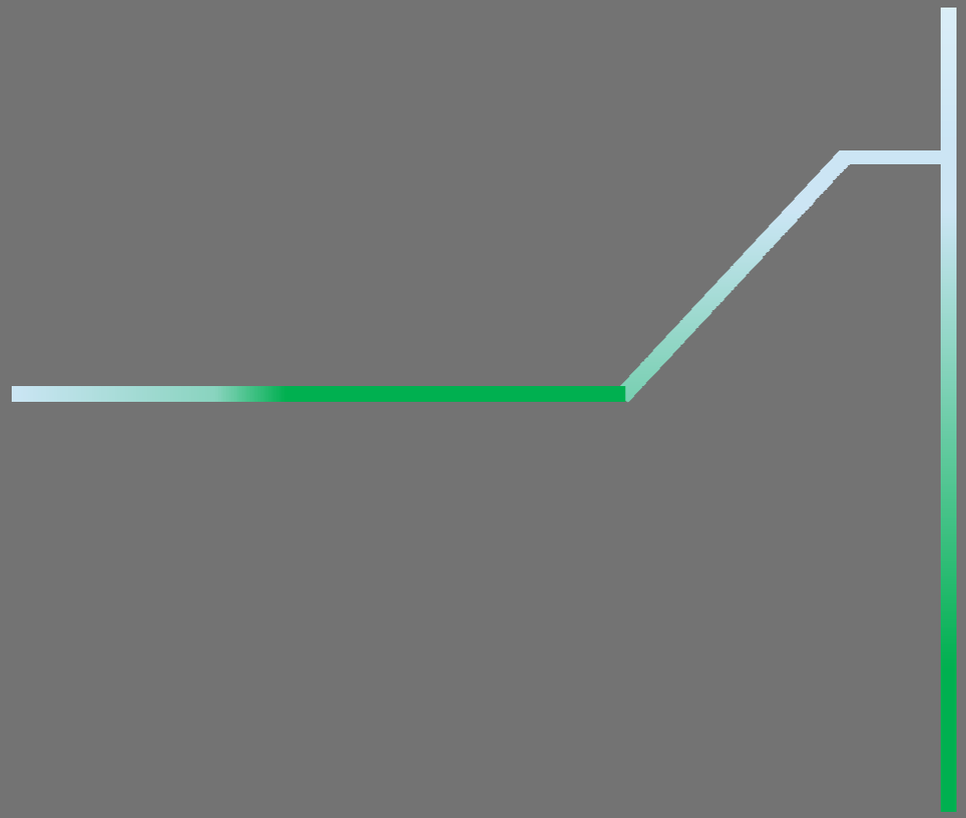
New in WS2025

Early Roadmap

Areas of Investigation

New to WS 2025

SMB in Windows and Windows Server 2025

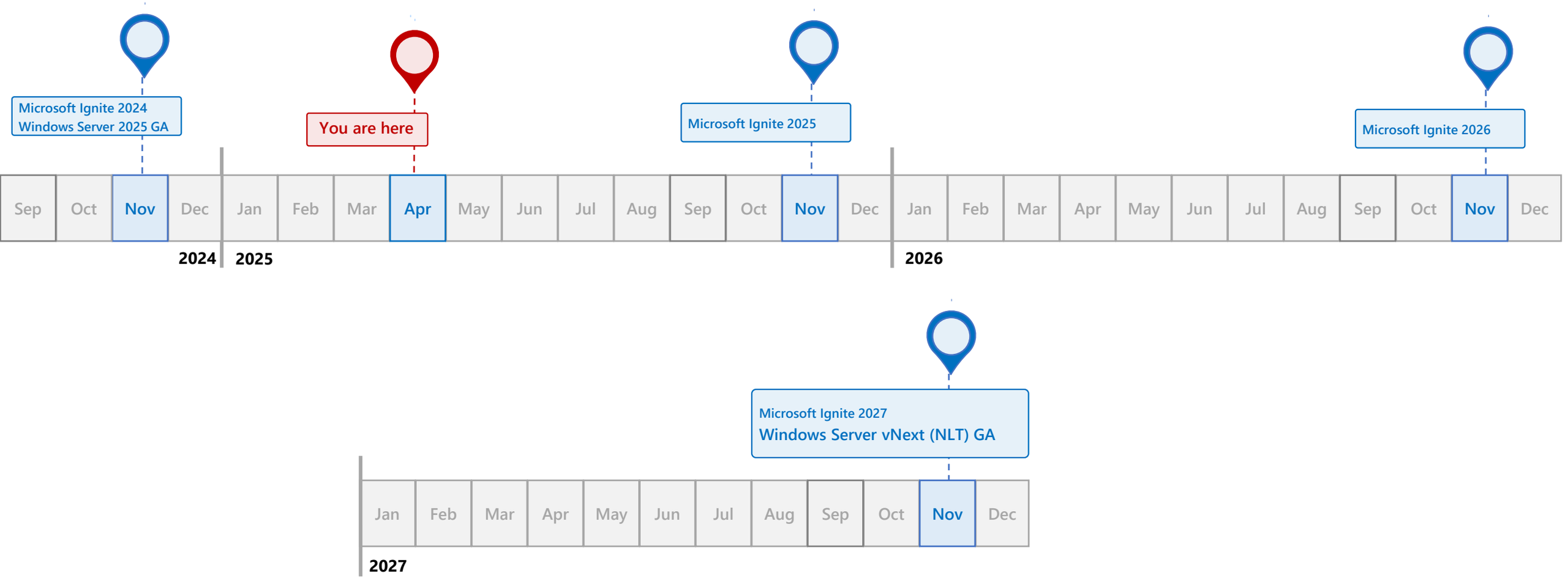


Next major release

SMB guest auth off in Pro
SMB global encrypt from client
SMB server dialect control
SMB signing required
SMB auth rate limiter
SMB NTLM disable option
SMB over QUIC all server editions
SMB over QUIC client access control
SMB alternative ports
SMB firewall rule tighten
SMB Mailslots disabled

Upcoming/Roadmap

Feature Milestones



SMB transport layer: limitations and inspiration

- Unexploited opportunities in many transfer patterns
  - Spatial locality, e.g. sequential writes
  - Temporal locality: repeat writes to single region
- **Proposal**: extended use of RDMA *across* I/Os
  - Wider registrations to exploit spatial locality
  - Longer-living registrations to exploit temporal locality
  - XMR: eXtended Memory Registrations
- **Proposal**: batched I/O
  - Better amortization of overhead of remote FS stack

SMB-over-RDMA today

- Using either inline RDMA SEND (small writes < 16 KiB)
- or, client exposes transfer buffer over RDMA; server acts on it
  - cached/double-buffered page pinned to physical memory for duration of I/O
  - require double-buffering for content stability/invariance
- Costs **per I/O**:
  - Memory registration
  - Client message to server to act on MR (token and I/O parameters: offset+length); Server performs RDMA READ or WRITE
  - Memory deregistration
- Using FRMRs (Fast-Register Memory Registrations) preallocated ahead of time to boost performance

SMB-over-RDMA today: DDP path

(DDP = Direct Data Placement)

Client: steps to a single write I/O

C1: If I/O size ≥ RdmaWriteThreshold, use memory registrations

C2: double buffer, if necessary

C3: lock/pin in physical memory

C4: register memory w/RNIC using FRMR

C5: RDMA SEND w/reference to MR ("memory descriptors" aka MrToken)
[-- SMB2 header --][-- WRITE command **w/MrToken** --]

→

Server

S1: RdmaRecvEventCallback: use MrToken to issue RDMA READ to client

S2: prepare destination (private buffer, or OS FS cache page)

S2: issue RDMA READ to pull payload from client

←

RNIC HW: payload flows from client to server buffer

→

S3: RdmaReadCallback: if data pulled to private buffer, must transfer to local FS stack

S4: signal result of local FS stack WRITE back to client by issuing RDMA SEND w/Remote Invalidate

S4: issue RDMA SEND w/Remote Invalidate of MrToken

←

C4: clean up: deregister memory, unpin/unlock, free double buffer (if any)

C5: end of I/O transfer

SRV MR management: concepts

- **Primitives** to allow client to create RDMA registrations on *server*
- Allows client to self-manage RDMA MRs to optimally fit its workload characteristics
  - Client has foresight, can build and execute plan
- An MR requires physical page backing
  - Current implementation requires DAX filesystem
  - DAX FS exposes physical extents of file
  - Physical extents are de facto bus-addressable regions that look like RAM
  - Future beyond DAX: “MDL Write”-like cache windows

SRV MR management: overview of FSCTLs

- Two FSCTLs that can be issued against a file object
- FSCTL_**CREATE**_MR:
  - Input: FileOffset, Length, access rights (read/write)
  - Op: SRV uses DAX QUERY_DIRECT_EXTENTS to obtain physical range
  - Output: server returns MR ID and NDK MR token
- FSCTL_**RELEASE**_MR:
  - Input: MR ID
  - Op: SRV performs security checks; frees MR
  - Output: status code (if security check passed, this should be infallible)

SMB client handling of SRV MR FSCTLs

- App can issue CREATE_MR and RELEASE_MR FSCTLs against a file handle
- SMB client introspects FSCTLs issued and responses received
- SMB client maintains ranges known to have a direct server mapping
- If I/O intersects a direct-mapped range, SMB client issues RDMA READ/WRITE against server
  - Aka "PUSHMODE I/O"
  - See past SNIA SDC talks by Tom Talpey, Matthew George
  - SDC 2017: *Remote Persistent Memory - With Nothing But Net - Tom Talpey* https://youtu.be/CahHW50SNjQ
  - SDC 2019: *Storage RDMA Push Mode to Persistent Memory via SMB3* https://youtu.be/E7zN8reyfXI (last character is a capital i)

PUSHMODE I/O w/primitives

Client

C1: issue CREATE_MR to server against an open file

C2: introspect CREATE_MR response; update internal books (list of direct-mapped MRs for file)

C: issue WRITE I/O

C: if I/O range intersects a direct-mapped range, issue RDMA WRITE directly to SRV

RNIC HW: payload flows from client to server buffer

C: else: flow through to non-PUSHMODE path (use inline SEND, or client registrations)

C: issue READ I/O

C: if I/O range intersects a direct-mapped range, issue RDMA READ directly from SRV

RNIC HW: payload flows from server to client

Server

S1: CREATE_MR FSCTL handler

S2: query direct access extents to file for requested slice

S3: create MR (FRMR or not) atop of slice

S4: internal book-keeping: record active MR for session/socket

S5: issue CREATE_MR response w/MrToken

SRV MR management: demo

- Demo

SRV MR management: gotchas and future steps

- MR registrations not unbounded
  - RNIC address translation table has limited space
  - Server unaware of writes through an MR; cannot expire/evict unused MRs
- DAX filesystems are rare in practice: requires specialized storage hardware
- Alternatives to DAX: local FS cache page backing
  - OS FS cache page lifetime dictated by OS Cache Manager (available RAM, memory pressure) not a remote client
- Write-through and flush
  - Server unaware of writes through an MR; must be explicitly signaled to write-through/flush
  - No efficient mechanism today; client must issue explicit command
- No SMB transform support (in a trival "PUSMODE I/O" mode)
  - Server unaware of writes; cannot de-transform (decrypt or decompress) payload
- RNIC adapter affinity: incompatible with SMB Multichannel

Batched I/O

- **Problem**: costly overhead of traversing the Windows filesystem stack before the bits ultimately make it to the wire
  - Filesystem minifilters: anti-virus and monitoring products, typically heavy
  - Lock acquisition
  - Conflict checks (arithmetic byte-range based checks: not constant time)
- **Solution**: provide BATCHED_WRITE and BATCHED_READ services; apps can issue a batch of I/O in a single syscall
  - Amortize overhead across a batch of I/Os for same file
  - Create opportunities for further optimization

Batched I/O is similar to:

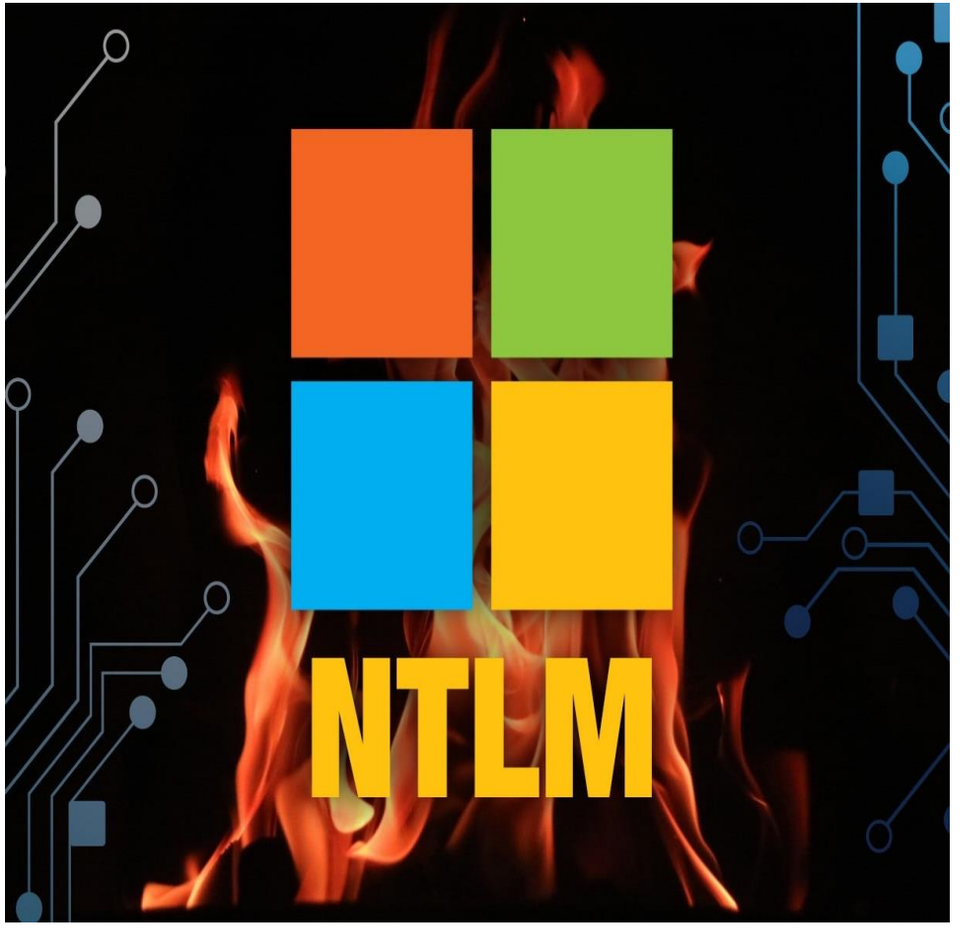
- **Bypass I/O**: high-performance I/O bypassing filters (though they can veto bypass'ed reads) on NVMe devices; introduced in Win11
  - Highlights performance cost of minifilters in stack
  - Minifilters will require enlightenment of new FSCTL_BATCHED_READ/WRITE controls
- **WriteFileGather**: input is an array of buffers, written contiguously
  - A batched write can be thought of WriteFileGather***Scatter***: takes an array of independent buffers, and scatters them across *independent* offsets in destination file

Batched I/O: development status

- Remote FS client stack: consolidate labour
  - Lock/resource acquisition
  - Conflict checks: oplocks, byte-range locks
- Under development
  - Transport layer batching
  - Tighter coupling w/RDMA
- Promotion to generic Windows FS stack call

Block SMB NTLM for a Windows application

- Application specific block of NTLM in the Internet Zone
- Granular mitigations without globally enabling block NTLM
- Opt-ins for specific processes:
  - Introduction of a New Process Mitigation Policy
  - Use SetProcessMitigationPolicy() to establish the policy
- Grant exception to server using existing PS Cmdlet
  - Set-SmbClientConfiguration -BlockNTLMServerExceptionList <input_list>
- Requires modification of applications



Area of Investigations

MFA and POSIX Extension

- Multi-Factor Authentication for SMB
  - Motivation: Address security concerns and comply with government regulations.
  - MFA solutions being evaluated:
    - Windows Hello for Business (WHfB)
    - Azure Entra ID
  - Exploring potential integration with third-party MFA solutions in the future.
- POSIX Extension
  - POSIX unlink and rename semantics supported by Windows local file system
  - POSIX mode bits
- Next Step
  - No Specific Timeline
  - Gathering feedback from the Community

Questions?